DAA/ LANGLEY
NAG1-582
IN-32
63716 31P.

TR-86-30

MARYLAND

TECHNICAL
RESEARCH
REPORT

A User's Guide for the Signal Processing Software for Image and Speech Compression Developed in The Communications and Signal Processing Laborator (CSPL) Version 1

by

P. Kumar, F.Y. Lin
V. Vaishampayan, and N. Farvardin

Electrical Engineering Department
and
Systems Research Center
University of Maryland
College Park, Maryland 20742

SYSTEMS RESEARCH CENTER

UNIVERSITY OF MARYLAND

COLLEGE PARK, MARYLAND 20742

A User's Guide for the Signal Processing Software for Image and Speech Compression Developed in The Communications and Signal Processing Laboratory (CSPL)

Version 1

by

P. Kumar, F. Y. Lin, V. Vaishampayan, and N. Farvardin

Electrical Engineering Department
and
Systems Research Center
University of Maryland
College Park, Maryland 20742

Abstract

In this report we provide a complete documentation of the software developed in the Communications and Signal Processing Laboratory (CSPL) during the period July 1985 - March 1986. Utility programs and subroutines that are developed for a user-friendly image and speech processing environment are described. Additional programs for data compression of image and speech type signals are included. Also, programs for the zero-memory and block transform quantization in the presence of channel noise are described. Finally, several routines for simulating the performance of image compression algorithms are included.

[1] This work was supported in part by a grant from NASA, Langley Research Center under grant NAG-1-582, in part by Martin Marietta Laboratories and in part by the National Science Foundation under NSFD CDR-85-00108.

I. Introduction:

This is the first of a series of technical reports issued by the *Communications and Signal Processing Laboratory (CSPL)* in the Electrical Engineering Department at the University of Maryland. The purpose of this report is to provide a complete and easy-to-read documentation of the software that has been developed in the CSPL during July 1985 - March 1986. Since the development of software packages supporting our research activities is an ongoing process in the CSPL, this technical report will be updated regularly (approximately twice a year) in an effort to reflect the future changes, corrections and expansions. We feel that the availability of a user-friendly software system and an appropriate documentation for it is extremely important in minimizing potential wastes of time and effort of the CSPL users.

In what follows we present a brief description of the facilities in the CSPL, the sources of funding and the organization of the software system.

A.) History:

The Communications and Signal Processing Laboratory (CSPL) was established in the Electrical Engineering Department at the University of Maryland in the academic year 1985-86. The aim of this laboratory is to provide the facility for experimental research in the general area of Communications and Signal Processing with dedicated facilities for image and speech processing. The Engineering Research Center (ERC) at the University of Maryland has played an instrumental role in establishing this laboratory by providing most of the funds for the purchase and maintenance of the equipment. The college of Engineering at the University of Maryland, as well as an equipment grant from the National Science Foundation have also made contributions to the purchase of equipment for the CSPL. Later on, the NSF-funded Systems Research Center has made additional contributions to the CSPL in the form of equipment funds and software support.

The research projects in the CSPL which are related to image and speech compression are primarily supported by NASA Langley Research Center, Martin Marietta Laboratories and the National Science Foundation through the Systems Research Center.

B.) Description of the Environment:

The heart and soul of the computational machinery in the CSPL is the Masscomp MC-500 DP. This is a dual processor 32 bit general purpose computer. Its features include two Motorola 68010 CPU's, two floating point processors, an array processor, and a complete data acquisition unit (which will be described in detail later) which includes an IEEE-488 bus interface. The peripherals include two graphics subsystems, an ethernet board, a 474 Megabyte and an 87 Megabyte Fujitsu Eagle Winchester Disks, a Storage Technology 2922 Tape Drive and an International Imaging Systems (IIS) System 575 image processing hardware.

The MC-500 DP is configured with five megabytes of memory. The operating system is Real Time Unix tm V2.2A. This is basically a port of AT&T Unix System III & V with some of the 4.2 BSD (Berkeley Standard Distribution of Unix) improvements and other changes to the file system for real time applications. The operating system has been optimized for fast I/O. This is the main reason why this machine was purchased. Also, since the rest of the machines in the Electrical Engineering Department and the Systems Research Center are also running Unix, software written on the Masscomp, that is not hardware dependent, is easily ported to the other machines.

The two most important peripherals are the IIS image processing unit and the data acquisition system. The IIS image processing unit is a stand alone system. That is, it has its own CPU, memory and video controllers, color monitor, camera and digitizer. It is connected to the host via an interface card in the Masscomp. It uses the Masscomp basically as an I/O device. The data acquisition system on the other hand is very much a part of the Masscomp. It is directly controlled by a Motorola 68010. It contains an IEEE 488 bus interface, an array of fifteen clock outputs, a four channel, high speed, twelve bit A/D (capable of sampling rates up to 1 MHz), an eight channel sixteen bit D/A, and a four channel S/H (sample and hold). The IEEE 488 bus interface can be used to control instruments in the laboratory. One of the applications of this is the control of a CAMAC crate which contains 16 K bytes of memory, and a function generator. A future application of the IEEE 488 is the control of a Matrix QCR digital camera.

As far as images are concerned, we have to deal with two different formats. The IIS images have a certain format and the Masscomp's graphics processors have a slightly different one. The Masscomp will display plain raster data. Each pixel can be nine bits deep. This allows us to use up to 512 different hues at one time. The IIS on the other hand can handle plain raster images, and it also has its own format where the images are prepended with a 512 byte header. This header contains information about the image itself and how it was saved/acquired etc. The IIS is also much more powerful than the Masscomp in the sense that each of the pixels can be up to 24 bits deep. This allows us to use over sixteen million different hues. So, in general, we cannot display color images that are generated by the IIS on the Masscomp. However, we can display grey level images and since the color images on the IIS are just a concatenation of three eight bit grey level images (one for each red, green, and blue) we can display one grey level at a time.

C.) Organization:

As indicated earlier, the main purpose of this report is to keep the interested community abreast of the facilities and especially the software system developed in the CSPL. In the next section of this report we will describe the different programs and subroutines that have been developed here. However, before describing these there are a few important issues that need to be mentioned.

1. The software described in this report is essentially supporting the research activities related to various problems in image and speech coding. What has been included here is only a subset of the software developed in the CSPL, the part which, we felt, is of interest to a typical user of the CSPL facility.

2. In developing the software described here, we have made an effort to be careful about compatibility and consistency. In other words, the output of one program can be used with little (if any) modifications as the input to another program, assuming that the input-output relationship is a logical one.

3. Our aim in developing this software is to help simulate the performance of a versatile communication system by individual building blocks.

4. This software will be expanded constantly. The documentation for it will be updated in subsequent technical reports.

In preparing this report, we have attempted to separate our software into one of four different categories, namely, (i) *Image Handling Routines,* (ii) *Speech Handling Routines,* and (iii) *Data Compression Routines.*

In the next section a table of contents is provided in which a list of the programs in each of the above categories is included. This is followed by Section III in which the complete documentation of individual programs is provided.

II. List of Software:

A.) Image handling software:

This software consists of a set of routines to help in the handling, manipulation and the display of images. These routines are all designed to be run on the Masscomp. Detailed information on these routines is found in the next section of this document, under subsection III. A.

1.) Stand alone executable routines

display.................	image display routine
iis.........................	switch to the iis environment
invert...................	negate an image
noise....................	make a noise file to simulate noisy channel
pad......................	convert an image less than 512x512 to 512x512
setgrey.................	set grey level color map
stats.....................	calculate the statistics of an image

2.) Subroutine Library:

These are subroutines which can be called only by FORTRAN routines (due to FORTRAN's column major array format). They make up a subroutine library called libimage_f.a. The manual page for these routines is in the next subsection III. A. under **libimage_f.a.**

rimage...................	read an image from a disk file into an array
rfimage..................	read a floating point image into an array
wimage..................	write an image from an array to a disk file
wfimage.................	write a floating point image into a disk file
msqerr...................	compute error statistics between two images

B.) Speech handling routines:

There is only one routine in this category. We have recently started work on our speech related software.

ad....................	speech acquisition (A-to-D) program
da....................	speech output (D-to-A) program

C.) Data Compression Related Software:

This software includes a set of programs that are used in developing various data compression and/or combined source-channel coding algorithms for image coding situations. Details can be found in subsection III.C.

assign	bit assignment algorithm for transform coding
channel	program for simulating a binary symmetric channel
coqd	channel-optimized quantizer design program
iqimag	program for decoding the output of the channel in 2D-DCT
qimag	program for quantizing the 2D-DCT transform coefficients
twodct	two-dimensional discrete cosine transform
twodct_q	uniform-threshold quantization of 2D-DCT coefficients
twodct_q_d	program for designing optimum uniform-threshold quantizers
twoidct	inverse two-dimensional discrete cosine transform

NAME

display – image display routine

SYNOPSIS

display [-I] [-r nrows -c ncols or -n dim] file

DESCRIPTION

Display will display both color and grey level integer images on the Masscomp 19 inch color monitor. The range of grey levels it can handle are 0-255. There are 9 bit planes for color and 1 bit plane for text.

The *-n* option assumes that the image is an IIS image and skips the first 512 bytes. In an IIS image the first 512 bytes are part of the header. Also, if an absolute path is not defined (i.e., a file name that does not begin with "/" or "~ "), then it looks for a file by the name of */usr/s575/Test/file.*

The -n option assumes the image is square. If it isn't, specify the number of rows and columns by using the -c and the -r options, respectively.

Also, if neither the *-n* nor the *-r*, *-c* options are used, the file is assumed to be square and ncol and nrow are assumed to be the square root of the size of *file*.

If the color map is not set for grey images, and a grey image is displayed, an image with a very distorted color map, resembling the original will appear on the monitor. It is very easy to fix this; just run the setgrey command in /usr/local/image.

AUTHOR

Praveen Kumar (it was rewritten)

FILES

None

SEE ALSO

setgrey(1)
The Masscomp Graphics Programming Manual
Programs in /usr/local/image
The system image library is /usr/lib/image

DIAGNOSTICS

Self-explanatory

BUGS

No known bugs

NAME

iis – liason routine between the Masscomp and the IIS system

SYNOPSIS

iis [-h] file1 [file2]

DESCRIPTION

Iis is a routine which helps in the display and conversion of files from the Masscomp environment to the IIS environment.

If no arguments are given to *iis*, it will change directories to /usr/s575/Test and start up the IIS Command Interpreter (CI). If both file1 and file2 are given as arguments to *iis*, it will copy file1 to /usr/s575/Test/file2, change directories as above, and start the CI. The argument file1 must be the name of a Masscomp type 512x512 image. If not, the IIS will not be able to display it (to pad pad smaller images, see pad). If no arguments are given, then it will just change directories and start the CI. If -h is an argument it will print a usage message.

Note: file2 must be less than six characters long and must be all capitals. This is a limitation of the IIS system.

Example:
To display an image file (e.g. GIRL) on the Masscomp on the IIS, one needs to enter the following series of commands:

```
pad -n 256 GIRL image.1
iis image.1 IMAGE1
```

When the command interpreter's prompt appears, enter:

```
> recover (file='IMAGE1' numlabels=0) > IMAGE1
```

This command is to record IMAGE1 as an image in the IIS image library.

AUTHOR

Praveen Kumar

FILES

None

SEE ALSO

su (1)
Programs in /usr/local/image
The system image library is /usr/lib/image

BUGS

None, so far.

NAME
invert – image manipulation program

SYNOPSIS
invert [-n dim] file1 file2

DESCRIPTION
Invert is used to change negative images into positives and vice versa. It reads a Masscomp type image (a plain raster file) of size (dim x dim) in file1, XORs (exclusive or) each byte with octal 0377, and puts the result in file2. The image in file2 is the negative of the image in file1. Because of the fact that XOR (XOR (foo)) = foo, inverting an image twice, gives the original image back.

AUTHOR
Praveen Kumar

FILES
None

SEE ALSO
Programs in /usr/local/image
The system image library is /usr/lib/image

DIAGNOSTICS
Self-Explanatory

BUGS
None so far

NAME

noise – image handling routine

SYNOPSIS

noise –n dim –p prob

DESCRIPTION

Noise generates a (dim x dim) noise image file. This file will contain a random sequence of bits. Each bit will have a *prob* probability of being a 1. This file XORed (exclusive or) with an image will simulate a noisy channel transmission of the image. The name of this file will be "cherr.prob". **WARNING – noise will erase old copies of a "cherr.*" file, if they exist.** The "prob" argument determines the probability of an error of a particular bit.

AUTHOR

Praveen Kumar

FILES

cherr.*

SEE ALSO

Programs in /usr/local/image
The system image library is /usr/lib/image

DIAGNOSTICS

Self explanatory

BUGS

None so far

NAME

pad – image conversion program

SYNOPSIS

pad [-n dim] or [-r nrows -c ncols] file1 file2

DESCRIPTION

Pad makes an image smaller than 512x512 (file1) into a 512x512 image (file2) by padding it with zeros. This program is necessary because the IIS image processing system will not accept foriegn images unless they are 512x512. So, if any image not 512x512 and not native to the IIS environment needs to be displayed on the IIS monitor, this program must be used to convert the image.

The -n option assumes the image in file1 is square. If it isn't, specify the number of rows and columns by using the -c and the -r options.

AUTHOR

Praveen Kumar

FILES

/usr/lib/image/*

SEE ALSO

iis(1l)
Programs in /usr/local/image
The system image library is /usr/lib/image

DIAGNOSTICS

Self explanatory

BUGS

None so far.

SETGREY
setgrey – image display program

SYNOPSIS
setgrey [# of levels]

DESCRIPTION
Setgrey is a program that assigns a grey scale to the color map from registers 0 to 255. If a numeric argument is specified, setgrey will map the scale through that register number and assign the white value (255) to the remaining registers.

AUTHOR
David Hsu

FILES
None

SEE ALSO
display(1) The Masscomp Graphic Programming Manual
Programs in /usr/local/image
The system image library is /usr/lib/image

DIAGNOSTICS
Self-explanatory

BUGS
None

NAME
stats – image analysis routine

SYNOPSIS
stats [-n dim] file

DESCRIPTION
Stats computes the mean, the variance, the standard deviation, the horizontal and vertical correlation coeffecient of a (dim x dim) Masscomp (plain raster) image.

AUTHOR
Praveen Kumar

FILES
None

SEE ALSO
Programs in /usr/local/image
The system image library is /usr/lib/image

DIAGNOSTICS
Self explanatory

BUGS
None so far

NAME

libimage_f.a – image handling library

SYNOPSIS

f77 [compiler options] [FORTRAN files] -limage_f

DESCRIPTION

Libimage_f.a is a library of FORTRAN (only) callable subroutines which are used for I/O and the processing of images on a Masscomp MC500DP computer. This machine is used in conjunction with the International Imaging Systems (IIS) image processor. Since, both these machines are used heavily, these routines were written so that they can be used to work on either normal raster images or IIS images. The only limitation of these routines is that the images passed to them **must** be square.

The routines are:

call rimage (file, dimi, array, dima, type)

This routine reads an image from a disk file into array

file – is a C type string (i.e., a character array whose last element must be a 0) which contains the file name of the image.
dimi – is an integer containing the dimension of the image (e.g., 256)
array – is a short integer (integer*2) square array. The image will be read into this array.
dima – is an integer containing the dimension of the array
type – is an integer which is either 0 or 1. If the image is of IIS format, type should be 1. Otherwise, type should be 0.

call rfimage (file, dimi, array, dima)

This routine is identical to rimage, above, except that it reads floating point images. Also, note that there is no **type** parameter here since, these images cannot be displayed. They are useful in software and algorithm development.

call wimage (file, dimi, array, dima)

This routine writes an array containing an image into a disk file

file – is a C type string (i.e., a character array whose last element must be a 0) which contains the file name of the image.
dimi – is an integer containing the dimension of the image (e.g., 256)
array – is a short integer (integer*2) square array. This array should contain the image when the subroutine is called.
dima – is an integer containing the dimension of the array

call wfimage (file, dimi, array, dima)

this routine is identical to wimage except that it writes floating point images.

call msqerr (image2, image1, dimi, dima, dist, mean, var, snr, snrdb)

this routine will compute some statistical information between two images.

image1 – is a short integer (i.e., integer*2) square array containing a processed image.
image2 – is also a short integer square array (of the exact same dimensions) containing the original or benchmark image.
dimi – is an integer containing the dimension of the image.
dima – is an integer containing the dimension of the array from which the image will be read.
dist – is a floating point (i.e., real) variable in which the distortion difference between the two images is returned.
mean – is a floating point variable in which the mean of image1 is returned.
var – is a floating point variable in which the variance of image1 is returned.
snr – is a floating point variable in which the signal-to-noise ratio is returned
snrdb – is a floating point variable in which the signal-to-noise ratio in dB is returned

AUTHOR

Praveen Kumar

FILES

None

SEE ALSO

Programs in /usr/local/image
The system image library is /usr/lib/image

DIAGNOSTICS

Self-explanatory

BUGS

The dima must be the dimension of the array in the read image subroutines. You cannot dimension an array to be 512x512 and let dima be 256. This will **not** work. In other words, you cannot dimension an array and use only part of it. You must use all of it.

NAME

ad – speech digitizing programs

SYNOPSIS

ad [-c clk] [-f freq] [-s nsamples] [-w nsecs] [-v] file

DESCRIPTION

Ad can sample analog signals through the Masscomp's A/D unit. The options are:

-c *clk* lets the user specify which clock device to use (default is /dev/dacp0/clk0)

-f *freq* lets the user specify the sampling frequency (default is 8,000 Hz)

-s *nsamples* lets the user specify the number of points (default is 16,000)

-w *nsecs* number of seconds to wait before beginning the operation (default is 0)

-v turns on the verbose flag (default is off)

file this argument specifies which file *ad* should read from.

The channel used for the A/D operation will be channel 0. This is a temporary limitation. The software will change so that the user may specify any channel he wishes.

AUTHOR

Praveen Kumar

FILES

None

SEE ALSO

Programs in /usr/local/spch
/usr/local/spch/da
The system speech library is in /usr/lib/spch

DIAGNOSTICS

Self-explanatory

BUGS

The maximum number of samples that *ad* can handle is 500,000.

NAME

da – output digitized speech

SYNOPSIS

da [-c clk] [-f freq] [-s nsamples] [-w nsecs] [-v] file

DESCRIPTION

Da can ouput sampled analog signals through the Masscomp's D/A unit. The options are:

-c *clk* lets the user specify which clock device to use (default is /dev/dacp0/clk0)

-f *freq* lets the user specify the sampling frequency (default is 8,000 Hz)

-s *nsamples* lets the user specify the number of points (default is 16,000)

-w *nsecs* number of seconds to wait before beginning the operation (default is 0)

-v turns on the verbose flag (default is off)

file this argument specifies which file *da* should read from.

The channel used for the D/A operation will be channel 0. This is a temporary limitation. The software will change so that the user may specify any channel he wishes.

AUTHOR

Praveen Kumar

FILES

None

SEE ALSO

Programs in /usr/local/spch
/usr/local/spch/ad
The system speech library is in /usr/lib/spch

DIAGNOSTICS

Self-explanatory

BUGS

The maximum number of samples that *da* can handle is 500,000.

NAME

assign – Bit assignment algorithm.

SYNOPSIS

assign [< input file] [> output file]

DESCRIPTION

The program *assign* allocates bits for quantization of the L^2 transform coefficients that correspond to the outputs of the 2-D Discrete Cosine Transform which has been performed on an image block of size $L \times L$. The allocation is done so as to maintain a specified average transmission rate while minimizing the mean squared error distortion across a given channel. The program prompts the user for the distortion (across the channel) vs. rate performance of the quantizers to be used for encoding the transform coefficients. It uses this data and the variances of the transform coefficients generated by the program *twodct* which are stored in the file 'res_2dct_var' to converge to the optimal bit allocation. The algorithm is an incremental bit allocation procedure which assigns a bit to the coefficient for which the decrease in squared error distortion is the largest. It completes the allocation in exactly L^2 steps. The allocation is optimal if the distortion vs. rate performance of the quantizers is convex.

INPUT

The user will be prompted for the following :
1) block size – must be a power of two, less than or equal to 256.
2) average transmission rate – a positive rational number whose product with L^2 is an integer.
3) distortion figures – the distortion data (for bit rates with integer values from 1 to 8) for the quantizers to be utilized.

OUTPUT

1) The theoretical distortion attained with the optimal bit assignment is computed and sent to stdout.
2) The bit assignment map is written into the integer file 'res_bit_asgt'.

COMPILATION COMMAND

```
f77 -o assign assign.f -limage_f
```

WARNING

The block size that the program asks for must have the same value used when computing the 2–D DCT using *twodct*. Note also that this size is NOT the size of the Mandela block.

FILES

Source : Located in /usr/lib/image.
Input : res_2dct_var
Output : res_bit_asgt

SEE ALSO

Program source listing for a more detailed description.
Pages of this manual for *twodct, twoidct, iqimag, channel* and *qimag.*

REFERENCES

1) A.V.Trushkin, "Optimal Bit Allocation Algorithm for Quantizing a Random Vector," Problems on Information Transmission, pp. 156-161, Jan. 1982.

AUTHOR

V. Vaishampayan

NAME
channel – channel simulator.

SYNOPSIS
channel [< input file]

DESCRIPTION
The program *channel* is a channel simulation program. It uses the output file of the program *noise* and performs a bit–by–bit Exclusive–Or operation between the binary equivalents of the integer data stored in the file 'res_2dct_q' and the contents of the noise file. Data contained in 'res_bit_asgt' is used to mask off unused portions of each 8–bit word. The output file of this program is called 'res_2dct_qc' and is written out in integer format.

INPUT
The user will be prompted for the following :
1) image size – must be a power of two, less than or equal to 256.
2) block size – must be a power of two, less than or equal to the image size.
3) noise file name – output file name of the *noise* program.

OUTPUT
The output is automatically written into the file 'res_2dct_qc'.

COMPILATION COMMAND
```
f77 -o channel channel.f -limage_f
```

WARNING
The block size that the program asks for must have the same values asked for by the programs *twodct, twoidct, iqimag* and *qimag*. Note also that this block size is NOT the size of the Mandela block but is the size of blocks into which the source image was blocked off.

FILES
Source : Located in /usr/lib/image.
Input : res_2dct_q, res_bit_asgt
Output : res_2dct_qc

SEE ALSO
Program source listing for a more detailed description.
Pages of this manual for *twodct, twoidct assign, iqimag, qimag* and *noise.*

AUTHOR
V. Vaishampayan

NAME

coqd – channel–optimized quantizer design program.

SYNOPSIS

coqd [< input file] [> output file]

DESCRIPTION

The program *coqd* designs a quantizer having a specified bit rate, for a specific source distribution, and assigns codes to the outputs of the quantizer in such a way that the squared error distortion across a memoryless binary symmetric channel (BSC) with a given crossover probability is (locally) minimized. The user specifies the source distribution from the class of generalized gaussian distributions, the bit rate and the crossover probability of the BSC. Beginning from a set of user specified reconstruction levels the algorithm iterates between the following two tasks till the mean squared error decreases by an amount smaller than a user specified threshold.
1) It first determines the optimal quantization thresholds and code assignments so that the distortion is minimized.
2) It then fixes the code assignment and thresholds and obtains the optimal set of reconstruction levels.
The program has two operating modes.
In mode 1 the starting code assignment and reconstruction levels are generated internally whereas in mode 2 the program uses the values specified by the user. The user is prompted by the program to select the desired operating mode. The program writes and reads from files that have the same format so it is possible to start the program up from a file generated by the program itself. This is especially useful when the user desires to improve the tolerance on the distortion values without having to rerun the program from scratch.

INPUT

The user will be prompted for the following :
1) Desired bit rate – Must be an integer between 1 and 8.
2) Distribution type – A positive real number that selects the type of distrbution from the class of generalized gaussian distributions.
3) Mean – The desired mean of the source distribution.
4) Variance – The desired variance of the source distribution.
5) Crossover probability of the BSC.
6) Mode select – Whether to read in initial values of reconstruction levels and code assignments or to generate them internally.
7) Tolerance on the distortion computation – The program stops when two successive computations of distortion differ by less than this value.

OUTPUT

In addition to all the parameters supplied by the input (except the initial reconstruction levels and code assignment) the program sends the following to stdout.
1) Mean Squared Error for the optimal design.
2) Signal to noise ratio for the optimal design.
3) Optimal threshold levels.
4) Optimal reconstruction levels.
5) Optimal code assignment.
Note : Optimal is used in the sense of Locally optimal.

COMPILATION COMMAND

```
f77 -o coqd coqd.f -lcmlib
```

FILES

Source: Located in /usr/lib/image

SEE ALSO

Program source listing for a more detailed description.

REFERENCES

N. Farvardin and V. Vaishampayan, "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," Submitted to IEEE Trans. Inform. Theory for publication.

AUTHOR

V. Vaishampayan

NAME

iqimag – decoder routine.

SYNOPSIS

iqimag [< input file]

DESCRIPTION

The program *iqimag* is used to decode the contents of the file called 'res_2dct_qc' created by the program *channel*. The data in this file consists of an $N \times N$ matrix of integers that represents data received from the noisy channel. This matrix is blocked off into submatrices of size $M \times M$ (the Mandela block size). Each element of the (i,j)'th submatrix is decoded using data contained in the file 'des_quant', and the (i,j)'th position of the bit allocation map available in 'res_bit_asgt'. The decoded data is then denormalized to its original mean and variance using data available in 'res_2dct_m' and 'res_2dct_var' after which it is written out into 'res_2dct_qcqi' in floating point format. Note that if zero bits have been assigned to any element in the bit allocation map then that element is decoded to the mean value as contained in 'res_2dct_m'.

INPUT
The user will be prompted for the following :
1) image size – must be a power of two, less than or equal to 256.
2) block size – must be a power of two, less than or equal to the image size.

OUTPUT
The output is automatically written into the real valued file 'res_2dct_qcqi'.

COMPILATION COMMAND
```
f77 -o iqimag iqimag.f -limage_f
```

WARNING

The block size that the program asks for must have the same values asked for by the programs *twodct, channel, twoidct,* and *qimag*. Note also that this block size is NOT the size of the Mandela block but is the size of blocks into which the source image was blocked off.

FILES

Source : Located in /usr/lib/image.
Input : res_2dct_qc, res_2dct_m, res_2dct_var, res_bit_asgt and des_quant
Output : res_2dct_qcqi

SEE ALSO

Program source listing for a more detailed description.
Pages of this manual for *twodct, twoidct, assign, channel* and *qimag*.

AUTHOR

V. Vaishampayan

NAME

qimag – quantization routine.

SYNOPSIS

qimag [< input file]

DESCRIPTION

The program *qimag* is used to quantize a matrix of transform coefficients of the *twodct* program which the program assumes are available in the file 'res_2dct' arranged in the Mandela form. The matrix of size $N, \times N$ containing the 2-D Discrete Cosine Transform coefficients is blocked off into submatrices of size $M \times M$ (the Mandela block size). Each element of the (i,j)'th submatrix is first normalized to zero mean and unit variance using the information contained in the files 'res_2dct_m' and 'res_2dct_var' generated by the program *twodct.* Then it is quantized using the quantizer structure available in the file 'des_quant' whose bit rate is equal to the (i,j)'th element of the bit assignment map contained in the file 'res_bit_asgt'. Each quantized element is then encoded using a binary code of length equal to the bit rate of the (i,j)'th quantizer and the decimal integer equivalent of this code is written out into the file 'res_2dct_q' in integer format.

INPUT
The user will be prompted for the following :
1) image size – must be a power of two, less than or equal to 256.
2) block size – must be a power of two, less than or equal to the image size.

OUTPUT
The output is automatically written into the integer file 'res_2dct_q'.

COMPILATION COMMAND
```
f77 -o qimag qimag.f quantize.o -limage_f
```

WARNING

The block size that the program asks for must have the same value used when computing the 2–D DCT using *twodct.* Note also that this size is NOT the size of the Mandela block.

FILES

Source : Located in /usr/lib/image.
Input : res_2dct, res_2dct_m, res_2dct_var, res_bit_asgt and des_quant
Output : res_2dct_q

SEE ALSO

Program source listing for a more detailed description.
Pages of this manual for *twodct, twoidct, iqimag, channel* and *assign.*

AUTHOR

V. Vaishampayan

NAME

twodct – two-dimensional (2-D) Discrete Cosine Transform (DCT) routine

SYNOPSIS

twodct (The user will be prompted for inputs - see INPUT inside DESCRIPTION for detail)

DESCRIPTION

twodct is used to find the 2-D DCT of an image
1) read the image from /usr/lib/image/*
2) perform 2-D DCT on the image with specific block size
3) re-arrange the 2-D DCT coefficients into Mandela form (optional)

INPUT
1) name of an input image file (e.g., MOON)
2) block size of 2-D DCT in integer format (assuming no. of rows = columns)
3) output option ? (0=output file "res_2dct" will contain 2-D DCT in Mandela form,
1=output file "res_2dct" will contain 2-D DCT)
4) write more information ? (0=Yes, 1=No) (See DIAGNOSTICS for detail)

OUTPUT
1) mean of the whole image will appear on the screen
2) variance of the whole image will appear on the screen
3) 2-D DCT coefficients of the image (in Mandela form) will be written in the file "res_2dct" with floating point format
4) mean of the transform coefficients in matrix form will be written in the file "res_2dct_m" with floating point format
5) variance of the transform coefficients in matrix form will be written in the file "res_2dct_var" with floating point format

DISCUSSION
Size of the image =256 by default (assuming total no. of rows = columns), changes must be made inside the program with the parameter 'idm'

A 2-D L*L DCT block is calculated as follows:
1) 1-D DCT with L samples is applied to all rows of the block
2) then, 1-D DCT with L samples is applied to all columns of the resultant block from (1) above

REFERENCES
1) M. Narasimha and A. Peterson, "On the Computation of the Discrete Cosine Transform" , *IEEE Trans. Commun.* , Vol. COMM-26, pp. 934-936, June 1978.
2) J. Modestino, D. Daut and A. Vickers, "Combined Source-Channel Coding of Images Using the Block Cosine Transform", *IEEE Trans. on Commun.* , Vol. COMM-29, pp. 1261-1274, Sept. 1981.

COMPILATION COMMAND
```
f77 -o twodct twodct.f -lcmlib -lspp -limage_f
```

FILES

Program in /usr/local/image
Input : /usr/lib/image/*
Outputs : res_2dct, res_2dct_m, res_2dct_var

DIAGNOSTICS

Information containing results of each step along with the execution of the program can be found when input (4) is set to zero. (especially useful when size of the image is equal to 8 and block size of the 2-D DCT is equal to 4)

AUTHOR

Frank Y.C. Lin

NAME

twodct_q – quantization with a two-dimensional (2-D) Discrete Cosine Transform (DCT) routine

SYNOPSIS

twodct_q (The user will be prompted for inputs - see INPUT inside DESCRIPTION for detail)

DESCRIPTION

twodct_q is used to quantize the 2-D DCT coefficients of an image

1) subtract off the mean and then normalize (to one) the variance of 2-D DCT coefficients with the use of mean and variance read from files "res_2dct_m" and "res_2dct_var"
2) quantize the 2-D DCT coefficients with uniform-threshold quantizers, based on stepsizes read from the file "res_2dct_ss", and output levels, which are centroids of the neighboring thresholds, will be based on the corresponding probability density assumption (See INPUT (4) for detail)
3) denormalize the quantized 2-D DCT coefficients with the use of variance read from the file "res_2dct_var" and then add the mean read from the file "res_2dct_m"

INPUT

1) block size of 2-D DCT in integer format (assuming no. of rows = columns)
2) maximum number of levels for all quantizers in integer format (e.g., 195) (See DISCUSSION inside DISCRIPTION for detail)
3) accuracy of integration in floating point format (e.g., 1.0e-15) (See DISCUSSION inside DISCRIPTION for detail)
4) probability densities option ? (0=all transform coefficients have Gaussian probability densities,
   1=all transform coefficients except the first one, which is assumed to have a Gaussian probability density, are assumed to have Laplacian probility densities)
5) write more information ? (0=Yes, 1=N0) (See DIAGNOSTICS for detail)
6) file of a 2-D DCT in mandela form will be read from the file "res_2dct" in floating point format
7) mean matrix of the 2-D DCT coefficients will be read from the file "res_2dct_m" in floating point format
8) variance matrix of the 2-D DCT coefficients will be read from the file "res_2dct_var" in floating point format
9) stepsizes for optimum uniform-threshold quantizer in matrix form will be read from the file "res_2dct_ss" in floating point format

OUTPUT

1) entropy calculated from output of each quantizer will appear on the screen
2) average entropy calculated from the output of all quantizers will appear on the screen
3) quantized Mandela 2-D DCT blocks of the image will be written in the file "res_2dct_q" with floating point format

DISCUSSION

Size of the image =256 by default (assuming total no. of rows = columns), changes must be made inside the program with 'idm'

No. of levels for each quantizer depends on the "accuracy" read from input (4); the better the "accuracy", the more levels will be used for each quantizer. However, the largest number of levels for each quantizer can not be larger than the "maximum no. of levels" read from input (3).

COMPILATION COMMAND

```
f77 -o twoquant twoquant.f -lcmlib  -limage_f
```

FILES

Program in /usr/local/image
Inputs : res_2dct, res_2dct_m, res_2dct_var, res_2dct_ss
Output : res_2dct_q

DIAGNOSTICS

Information containing results of each step along with the execution of the program can be found when input (5) is set to zero. (especially useful when size of the image is equal to 8 and block size of the 2-D DCT is equal to 4)

AUTHOR

Frank Y.C. Lin

NAME

twodct_q_d – routine designing optimum uniform-threshold quantizers for quantization of the two-dimensional (2-D) Discrete Cosine Transform (DCT) coefficients which are assumed to have either Gaussian or Laplacian probability densities

SYNOPSIS

twodct_q_d (The user will be prompted for inputs - see INPUT inside DESCRIPTION for detail)

DESCRIPTION

twodct_q_d is used to find a set of optimum uniform-threshold quantizers (i.e., the stepsizes) for quantizing blocks of either Gaussian or Laplacian random variables (See INPUT (1))

INPUT

1) option ? (0=all transform coefficients have Gaussian probability densities
   1=all transform coefficients except the first one, which is assumed to have a Gaussian probability density, are assumed to have Laplacian probability densities)
2) block size of 2-D DCT in integer format (assuming no. of rows = columns)
3) variances of the transform coefficients in matrix form will be read from the file "res_2dct_var" in floating point format
4) average quantizer output entropy per sample in floating point format

OUTPUT

1) lagrange multiplier corresponding to the optimum scheme will appear on the screen (See reference [1] for detail)
2) normalized lagrange multiplier (to unit variance) will appear on the screen (See reference [1] for detail)
3) no. of levels, N, for each quantizer will appear on the screen
4) stepsize for each quantizer will appear on the screen
5) corresponding entropy and distortion of each quantizer based on the assumed probability densities will appear on the screen
6) matrix of stepsizes, upon which optimum-threshold quantizers are designed, will be written in the file "res_2dct_ss" with floating point format

DISCUSSION

Uniform-threshold quantization is employed to quantize transform coefficients which are assumed to have either Gaussian or Laplacian probability densities. Algorithm for optimum stepsize assignment among the quantizers is developed from reference [1].

REFERENCES

1) N. Farvardin and F. Y. Lin , "Performance of Entropy-Constrainted Block Transform Quantizers", submitted to IEEE Trans. Inform. Theory for publication. Also, Technical Report, TR-85-32, Systems Research Center, University of Maryland, College Park, MD, Jan. 1986.

COMPILATION COMMAND

```
f77 -o twodct_q_d twodct_q_d.f -lcmlib -limage_f
```

FILES

Program in /usr/local/image
Input : res_2dct_var
Output : res_2dct_ss

DIAGNOSTICS

Major results of each step of the algorithm will be found along with the execution. As a result, it is recommended to direct the outputs of the program into a file.

AUTHOR

Frank Y.C. Lin

NAME

twoidct – two-dimensional (2-D) Inverse Discrete Cosine Transform (IDCT) routine

SYNOPSIS

twoidct (The user will be prompted for inputs - see INPUT inside DISCRIPTION for detail)

DESCRIPTION

twoidct is used to find the 2-D IDCT of an image
1) re-arrange the Mandela blocks to 2-D IDCT (optional)
2) perform the 2-D IDCT

INPUT

1) input option ? (0=input file "res_2dct" will contain 2-D DCT in Mandela form,
   1=input file "res_2dct" will contain 2-D DCT)
2) file of a 2-D DCT (in Mandela form) will be read from the file "res_2dct_q" in floating point format
3) block size of 2-D IDCT in integer format (assuming no. of rows = columns)
4) write more information ? (0=Yes, 1=No) (See DIAGNOSTICS for detail)

OUTPUT

1) the 2-D IDCT result will be written in the file "image" with integer format

DISCUSSION

Size the image =256 by default (assuming total no. of rows = columns), changes must be made inside the program with the parameter 'idm'

A 2-D L*L IDCT block is calculated as follows:
1) 1-D IDCT with L samples is applied to all rows of the block
2) then, 1-D IDCT with L samples is applied to all columns of the resultant block from (1) above

REFERENCES

1) M. Narasimha and A. Peterson, "On the Computation of the Discrete Cosine Transform" , *IEEE Trans. on Commun.* , Vol . COMM-26, pp. 934-936, June 1978.
2) J. Modestino, D. Daut and A. Vickers, "Combined Source-Channel Coding of Images Using the Block Cosine Transform", *IEEE Trans. on Commun.* , Vol. COMM-29, pp. 1261-1274, Sept. 1981.

COMPILATION COMMAND

```
f77 -o twoidct twoidct.f -lcmlib -lspp -limage_f
```

FILES

Program in usr/local/image
Input : res_2dct_q
Output : image

DIAGNOSTICS

Information containing results of each step along with the execution of the program can be found when input (4) is set to zero. (especially useful when size of the image is equal to 8 and block size of the 2-D DCT is equal to 4)

AUTHOR

Frank Y.C. Lin